L Number	Hits	Search Text	DB	Time stamp
-	200	phuoc.xa.	USPAT	2004/02/13 09:39
-	8	("5805823" | "6006254" | "6078948" | "6195680" | "6253241" | "6272536" | "6275496" | "6345293").PN.	USPAT	2004/02/13 13:21
-	24	("5318450" | "5515098" | "5630103" | "5717923" | "5721827" | "5724521" | "5740549" | "5754939" | "5758078" | "5805153" | "5848396" | "5864823" | "5887133" | "5933603" | "5995943" | "6002393" | "6002394" | "6006241" | "6006257" | "6014701" | "6029045" | "6029195" | "6122658" | "6253241").PN.	USPAT	2004/02/13 09:24
-	14	proxy near coordinat$3	USPAT	2004/02/13 09:46
-	11	(proxy near coordinat$3) and @ad<19991118	USPAT	2004/02/13 09:40
-	633	proxy and isp	USPAT	2004/02/13 09:46
-	353	proxy and isp and bandwidth	USPAT	2004/02/13 09:47
-	101	(proxy and isp and bandwidth) and ((re near1 transmi$8) or retransmi$7)	USPAT	2004/02/13 09:48
-	82	((proxy and isp and bandwidth) and ((re near1 transmi$8) or retransmi$7)) and @ad<19991118	USPAT	2004/02/13 11:11
-	0	retransmi$7 near5 data near4 (other near3 users)	USPAT	2004/02/13 11:44
-	86	data near4 (other near3 users)	USPAT	2004/02/13 11:43
-	2	retransmi$7 with data with (other with users)	USPAT	2004/02/13 11:45
-	10	retransmi$7 same data same (other with users)	USPAT	2004/02/13 11:45
-	7295	full adj duplex	USPAT	2004/02/13 13:21
-	176	(full adj duplex) same retransmi$6	USPAT	2004/02/13 13:21
-	5	((full adj duplex) same retransmi$6) and client and server	USPAT	2004/02/13 13:26
-	543	(full adj duplex) same incoming	USPAT	2004/02/13 13:23
-	78	((full adj duplex) same incoming) same (transmi$6 or send$3) same (simultane$6)	USPAT	2004/02/13 13:24
-	34	((full adj duplex) same incoming) same (transmi$6 or send$3) near4 (simultane$6)	USPAT	2004/02/13 13:25
-	34465	(transmi$6 or send$3) near6 (simultane$6)	USPAT	2004/02/13 13:25
-	857	((transmi$6 or send$3) near6 (simultane$6)) and client and server	USPAT	2004/02/13 13:26
-	183	(((transmi$6 or send$3) near6 (simultane$6)) and client and server) and duplex	USPAT	2004/02/13 13:30
-	97	((((transmi$6 or send$3) near6 (simultane$6)) and client and server) and duplex) and download$3	USPAT	2004/02/13 13:29
-	299	(((transmi$6 or send$3) near6 (simultane$6)) and client and server) and (bidirection$3 (bi near1 direction$3) bidirection$3 duplex)	USPAT	2004/02/13 13:32
-	183	((((transmi$6 or send$3) near6 (simultane$6)) and client and server) and duplex) and duplex	USPAT	2004/02/13 13:30
-	299	(((transmi$6 or send$3) near6 (simultane$6)) and client and server) and ((((transmi$6 or send$3) near6 (simultane$6)) and client and server) and (bidirection$3 (bi near1 direction$3) bidirection$3 duplex))	USPAT	2004/02/13 14:11

-	1910	north$5.inv.	USPAT	2004/02/13 14:12
-	0	northeult.inv.	USPAT	2004/02/13 14:14
-	591	download$3 same (stor$4 memory) same forward$3	USPAT	2004/02/13 14:15
-	95	download$3 with (stor$4 memory) with forward$3	USPAT	2004/02/13 14:15
-	79	(download$3 with (stor$4 memory) with forward$3) and @ad<19991118	USPAT	2004/02/13 15:08
-	1	5938737.pn.	USPAT	2004/02/17 11:00
-	9	("5136291" | "5534913" | "5555377" | "5557749" | "5673322" | "5673392" | "5732216" | "5732219" | "5742773").PN.	USPAT	2004/02/13 15:08
-	6	5938737.URPN.	USPAT	2004/02/13 15:12
-	28554	(receiv$3 adj3 (data packet file)) same stor$3 same (forward$3 transmit$4 send$3)	USPAT	2004/02/17 11:02
-	23853	(receiv$3 near (data packet file)) same stor$3 same (forward$3 transmit$4 send$3)	USPAT	2004/02/17 11:02
-	8401	(receiv$3 near (data packet file)) with stor$3 with (forward$3 transmit$4 send$3)	USPAT	2004/02/17 11:05
-	5125	(client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 transmit$4 send$3)	USPAT	2004/02/17 11:06
-	1992	(client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 re-transmit$4 send$3)	USPAT	2004/02/17 11:06
-	176	((client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 re-transmit$4 send$3)) and full near duplex	USPAT	2004/02/17 11:07
-	165	(((client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 re-transmit$4 send$3)) and full near duplex) and @ad<19991118	USPAT	2004/02/17 11:08
-	2113203	((((client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 re-transmit$4 send$3)) and full near duplex) and @ad<19991118) and 709/2$$.ccls.c	USPAT	2004/02/17 11:08
-	32	((((client customer user computer) near4 (receiv$3 near (data packet file)) with (forward$3 re-transmit$4 send$3)) and full near duplex) and @ad<19991118) and 709/2$$.ccls.	USPAT	2004/02/17 12:13
-	454	adsl.ti.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 12:16
-	1	(adsl and (internet near access)).ti.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 12:14
-	1	adsl.ti. and Bergkvist.inv.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 12:22
-	358	adsl near2 transmission	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 12:22
-	20	adsl near2 transmission.ti.	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 12:26
-	3	(manipulat$3 chang$3 modif$7) adj6 (option near3 dhcp)	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:31

-	0	(manipulat$3 chang$3 modif$7) adj6 ((option near field) adj7 dhcp)	USPAT	2004/02/17 12:30
-	65	dhcp near5 option	USPAT	2004/02/17 12:31
-	32	(dhcp near5 option) and ISP	USPAT	2004/02/17 12:35
-	21	(dhcp near5 option) and ISP and $dsl	USPAT	2004/02/17 12:46
-	1	5790548.pn.	USPAT	2004/02/17 12:46
-	1	5790548.pn. and dhcp	USPAT	2004/02/17 12:47
-	0	5790548.pn. and dhcp near3 option	USPAT	2004/02/17 12:47
-	1	5790548.pn. and (chang$3 modif$7 manipulat$3) near3 dhcp	USPAT	2004/02/17 12:47
-	0	6285889.pn. and email	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:33
-	8	(email e-mail (electronic near mail)) with embed$4 with print$3	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:47
-	2168	(email e-mail (electronic near mail)) with print$3	USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/02/17 14:47
-	1141	(email e-mail (electronic near mail)) with print$3	USPAT	2004/02/17 14:49
-	1	5884031.pn.	USPAT	2004/02/20 16:51
-	1	5884031.pn. and (client with (stor$3 memor$3 buffer$3))	USPAT	2004/02/20 15:58
-	1	5884031.pn. and (information with (stor$3 memor$3 buffer$3))	USPAT	2004/02/20 16:01
-	1	5884031.pn. and (hardfile)	USPAT	2004/02/20 16:03
-	1	5884031.pn. and (forward$3 retransmit$4 transmit$4)	USPAT	2004/02/20 16:05
-	1	5884031.pn. and (rout$3 send$3)	USPAT	2004/02/20 16:09
-	1	5884031.pn. and download$3	USPAT	2004/02/20 16:55
-	1	5884031.pn. and after	USPAT	2004/02/20 16:29
-	3739	on-the-fly	USPAT	2004/02/20 16:29
-	306	on-the-fly near2 data	USPAT	2004/02/20 16:31
-	20	(on-the-fly near2 data) and 709/$$$.ccls.	USPAT	2004/02/20 16:30
-	1	5884031.pn. and time	USPAT	2004/02/20 17:01
-	0	5884031.pn. and track$3	USPAT	2004/02/20 17:31
-	0	5884031.pn. and distribut$3	USPAT	2004/02/20 17:02
-	0	5884031.pn. and monitor$3	USPAT	2004/02/20 17:02
-	1	5884031.pn. and broadcast$3	USPAT	2004/02/20 17:02
-	1	6334151.pn. and track$3	USPAT	2004/02/20 17:31
-	0	6334151.pn. and servers	USPAT	2004/02/20 17:31
-	1	5884031.pn. and servers	USPAT	2004/02/20 17:34
-	14	((forward$3 re-transmission retransmit$4) near5 (data stream$3 file packet)) with (storage near server)	USPAT	2004/02/20 17:36

US006334151B1

(12) **United States Patent**
Bolam et al.

(10) **Patent No.: US 6,334,151 B1**
(45) **Date of Patent: Dec. 25, 2001**

(54) **PUBLISH AND SUBSCRIBE DATA PROCESSING APPARATUS, METHOD AND COMPUTER PROGRAM PRODUCT WITH DECLARATION OF A UNIQUE PUBLISHER BROKER**

(75) Inventors: **Steven William Bolam**, Eastleigh; **Brian Clive Homewood**, Winchester; **Andrew Hickson**, West Wellow; **John Michael Knapman**, Eastleigh; **David Ware**, Romsey, all of (GB)

(73) Assignee: **International Business Machines Corporation**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/288,988**

(22) Filed: **Apr. 9, 1999**

(30) **Foreign Application Priority Data**

Dec. 23, 1998 (GB) 9828278

(51) **Int. Cl.**[7] **G06F 15/173**; G06F 15/16
(52) **U.S. Cl.** **709/225**; 709/244; 709/206
(58) **Field of Search** 706/62; 709/224, 709/206, 226, 252, 225, 244; 707/6, 1, 10; 345/330, 331, 332

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,815,030		3/1989	Cross et al.	364/900
5,136,708	*	8/1992	Lapourtre et al.	395/650
5,675,802	*	10/1997	Allen et al.	717/3
5,768,528	*	6/1998	Stumm	709/231
5,867,709	*	2/1999	Klencke	717/2
5,867,799	*	2/1999	Lang et al.	707/1
5,983,214	*	11/1999	Lang et al.	707/1
5,987,460	*	11/1999	Niwa et al.	707/6
5,999,975	*	12/1999	Kittaka et al.	709/224
6,014,654	*	1/2000	Ariyoshi	706/62
6,021,443	*	2/2000	Bracho et al.	709/241
6,154,781	*	1/2000	Bolam et al.	709/238
6,202,093	*	3/2001	Bolam et al.	709/225
6,240,451	*	5/2001	Campbell et al.	709/224

FOREIGN PATENT DOCUMENTS

806731 11/1997 (EP) .

OTHER PUBLICATIONS

Shan, Yen-Ping et al. "A multiple-platform multi-language distributed object-oriented messaging system", ACM Conference on Object Oriented Programming Systems Languages and Applications, pp. 27-29, Oct. 1993.*
Choy, D.M. et al. "Services and architectures for electronic publishing", IEEE Compcon '96, ISBN: 0-8186-7414-8, pp. 291-297, Feb. 1996.*
Cunningham, R.J. et al. "OSM: an Open Service Model for global information brokerage and distribution", IEEE Col. on Intelligent WWW Agents, Mar. 1997, pp. 2/1-2/5.*
Edge: Work-Group Computing Report, v7, p15(1), Oct. 14, 1996, "Internet Access: Intermind Communicator takes Web by storm . . . ".

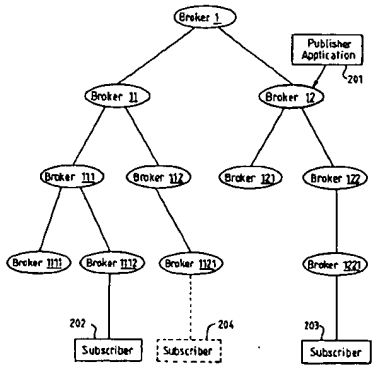
* cited by examiner

Primary Examiner—Mark H. Rinehart
Assistant Examiner—Jason D. Cardone
(74) *Attorney, Agent, or Firm*—Edward H. Duffield

(57) **ABSTRACT**

In a publish/subscribe data processing broker network having a plurality of broker data processing apparatuses each of which has an input for receiving published messages directly from a publisher application and/or receiving subscription data from a subscriber application, a first broker data processing apparatus has: a unit for receiving a data message published on a first topic by a first publisher application; and a unit for forwarding the received published data message to a subscriber application which has requested, by entering subscription data, to receive a message on the first topic; wherein the first broker data processing apparatus sends a declaration to at least one other broker data processing apparatus of said plurality of broker data processing apparatuses declaring that the first broker data processing apparatus is the only broker data processing apparatus that is directly communicating with a publisher application that is publishing on the first topic.

7 Claims, 2 Drawing Sheets



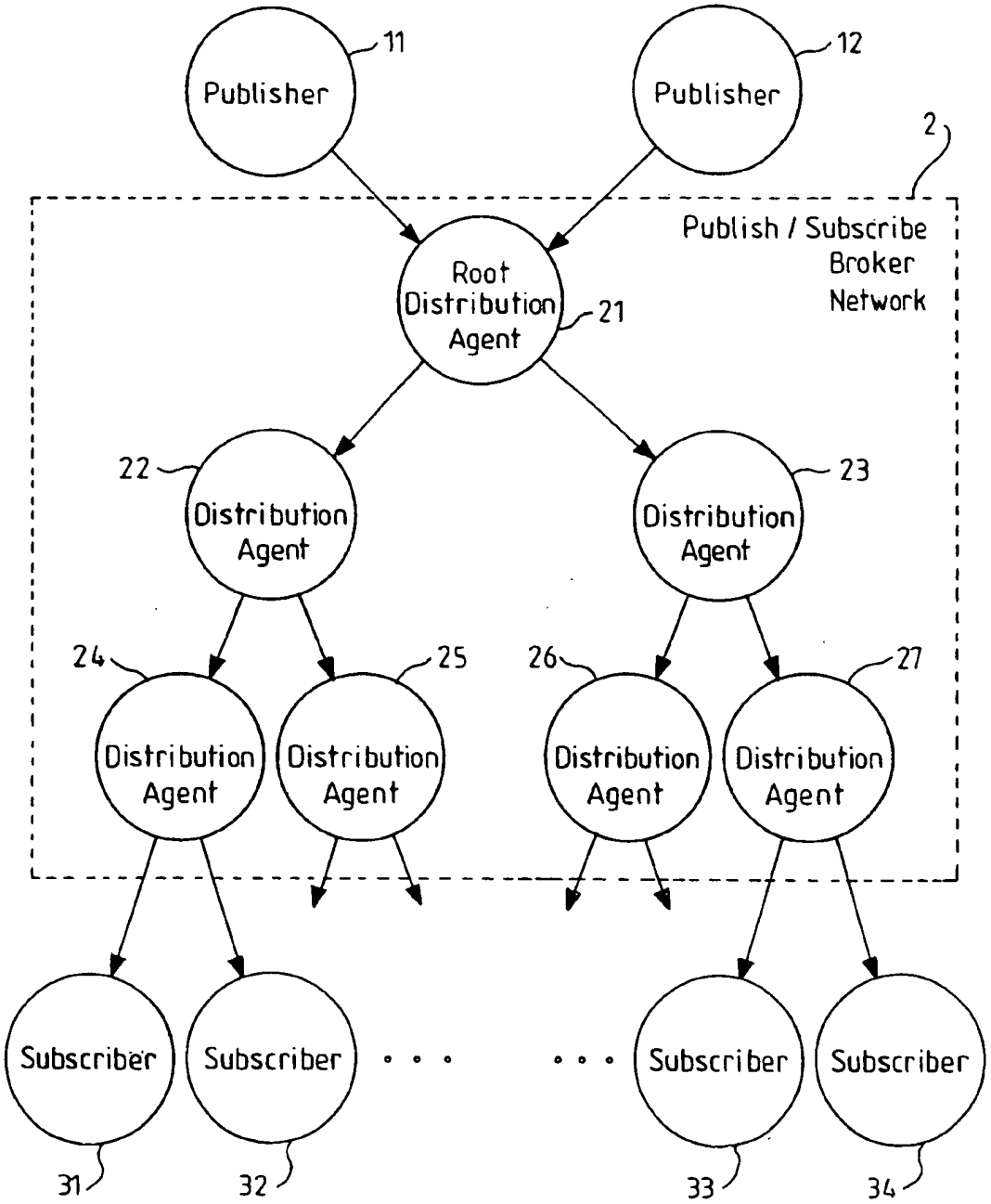

Publisher
11
Publisher
12
2
Publish / Subscribe
Broker
Network
Root
Distribution
Agent
21
22
Distribution
Agent
23
Distribution
Agent
24
25
26
27
Distribution
Agent
Distribution
Agent
Distribution
Agent
Distribution
Agent
Subscriber
Subscriber
Subscriber
Subscriber
31
32
33
34

FIG. 1

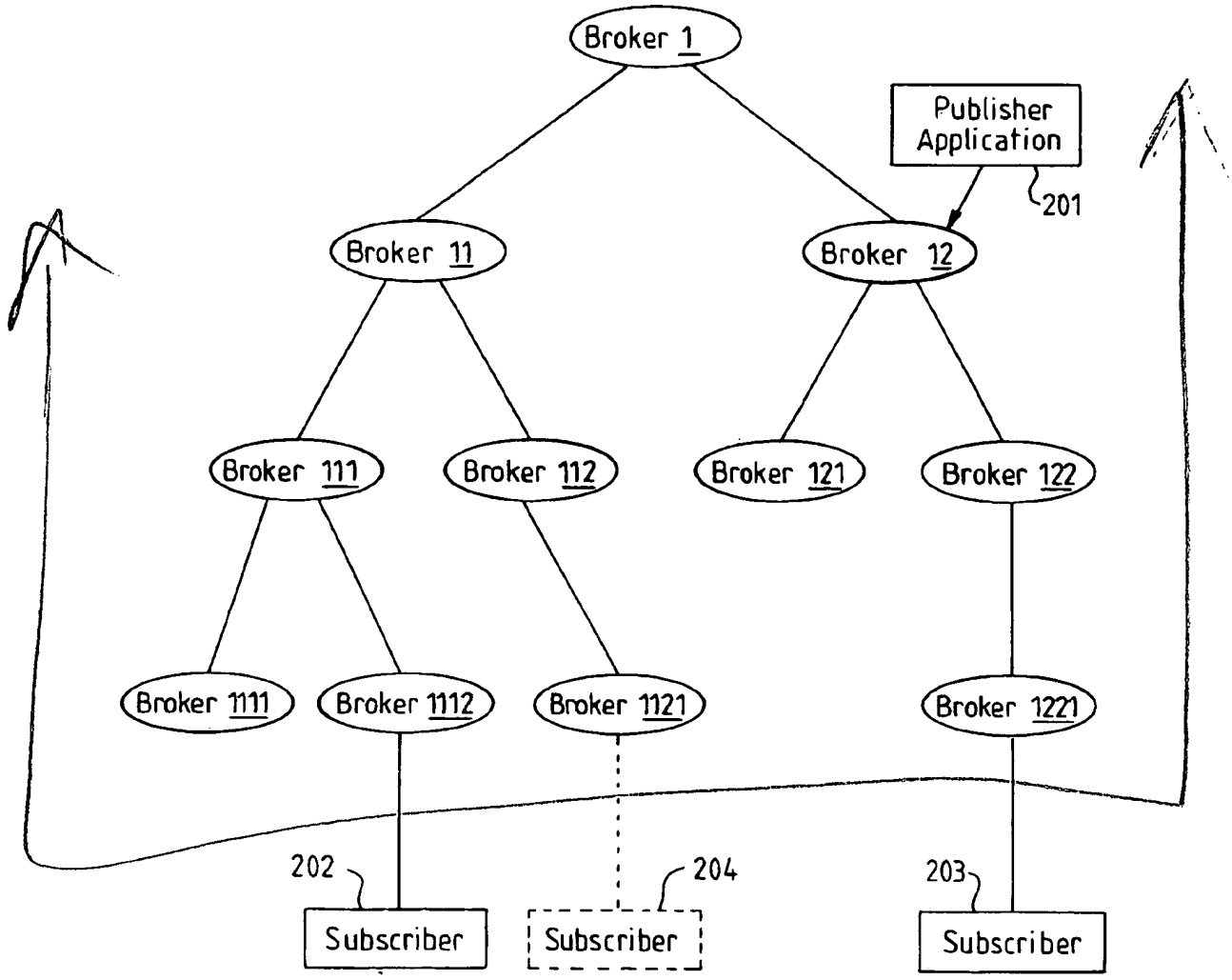

Broker 1
Publisher Application
201
Broker 11
Broker 12
Broker 111
Broker 112
Broker 121
Broker 122
Broker 1111
Broker 1112
Broker 1121
Broker 1221
202
204
203
Subscriber
Subscriber
Subscriber

FIG. 2

PUBLISH AND SUBSCRIBE DATA PROCESSING APPARATUS, METHOD AND COMPUTER PROGRAM PRODUCT WITH DECLARATION OF A UNIQUE PUBLISHER BROKER

FIELD OF THE INVENTION

The present invention relates to the field of data processing and more specifically to data processing which distributes messages from suppliers (called, hereinafter, "publishers") of data messages to consumers (called, hereinafter "subscribers") of such messages.

BACKGROUND OF THE INVENTION

Publish/subscribe data processing systems have become very popular in recent years as a way of distributing data messages from publishing computers to subscribing computers. The increasing popularity of the Internet, which has connected a wide variety of computers all over the world, has helped to make such publish/subscribe systems even more popular. Using the Internet, a World Wide Web browser application (the term "application" or "process" refers to a software program, or portion thereof, running on a computer) can be used in conjunction with the publisher or subscriber in order to graphically display messages. Such systems are especially useful where data supplied by a publisher is constantly changing and a large number of subscribers needs to be quickly updated with the latest data. Perhaps the best example of where this is useful is in the distribution of stock market data.

In such systems, publisher applications of data messages do not need to know the identity or location of the subscriber applications which will receive the messages. The publishers need only connect to a publish/subscribe distribution agent process (the terms "distribution agent" and "broker" are used interchangeably herein), which is included in a group of such processes making up a broker network, and send messages to the distribution agent process, specifying the subject of the message to the distribution agent process. The distribution agent process then distributes the published messages to subscriber applications which have previously indicated to the broker network that they would like to receive data messages on particular subjects. Thus, the subscribers also do not need to know the identity or location of the publishers. The subscribers need only connect to a distribution agent process.

One such publish/subscribe system which is currently in use, and which has been developed by the Transarc Corp. (a wholly owned subsidiary of the assignee of the present patent application, IBM Corp.) is shown in FIG. **1**. Publishers **11** and **12** connect to the publish/subscribe broker network **2** and send published messages to broker network **2** which distributes the messages to subscribers **31, 32, 33, 34**. Publishers **11** and **12**, which are data processing applications which output data messages, connect to broker network **2** using the well known interapplication data connection protocol known as remote procedure call (or RPC). Each publisher application could be running on a separate machine, alternatively, a single machine could be running a plurality of publisher applications. The broker network **2** is made up of a plurality of distribution agents (**21** through **27**) which are connected in a hierarchical fashion which will be described below as a "tree structure". These distribution agents, each of which could be running on a separate machine, are data processing applications which distribute data messages through the broker network **2** from publishers

to subscribers. Subscriber applications **31, 32, 33** and **34** connect to the broker network **2** via RPC in order to receive published messages.

Publishers **11** and **12** first connect via RPC directly to a root distribution agent **21** which in turn connects via RPC to second level distribution agents **22** and **23** which in turn connect via RPC to third level distribution agents **24, 25, 26** and **27** (also known as "leaf distribution agents" since they are the final distribution agents in the tree structure). Each distribution agent could be running on its own machine, or alternatively, groups of distribution agents could be running on the same machine. The leaf distribution agents connect via RPC to subscriber applications **31** through **34**, each of which could be running on its own machine.

In order to allow the broker network **2** to determine which published messages should be sent to which subscribers, publishers provide the root distribution agent **21** with the name of a distribution stream for each published message. A distribution stream (called hereinafter a "stream") is an ordered sequence of messages having a name (e.g., "stock" for a stream of stock market quotes) to distinguish the stream from other streams. Likewise, subscribers provide the leaf distribution agents **31** through **34** with the name of the streams to which they would like to subscribe. In this way, the broker network **2** keeps track of which subscribers are interested in which streams so that when publishers publish messages to such streams, the messages can be distributed to the corresponding subscribers. Subscribers are also allowed to provide filter expressions to the broker network in order to limit the messages which will be received on a particular stream (e.g., a subscriber **31** interested in only IBM stock quotes could subscribe to the stream "stock" by making an RPC call to leaf distribution agent **24** and include a filter expression stating that only messages on the "stock" stream relating to IBM stock should be sent to subscriber **31**).

The above-described publish/subscribe architecture provides the advantage of central coordination of all published messages, since all publishers must connect to the same broker (the root) in order to publish a message to the broker network. For example, total ordering of published messages throughout the broker network is greatly facilitated, since the root can easily assign sequence numbers to each published message on a stream. However, this architecture also has the disadvantage of publisher inflexibility, since each publisher is constrained to publishing from the single root broker, even when it would be much easier for a publisher to connect to a closer broker.

Accordingly, publish/subscribe software designers are beginning to consider architectures where publishers are allowed to publish messages directly to any broker in the broker network. This clearly has the advantage of removing the above-mentioned constraint on publishers. However, as with any tradeoff, it presents other problems. One of the major problems is that since a publisher can publish from any broker, subscription data (data indicating which subscribers have subscribed to which streams/topics) must be propagated throughout the broker network, as it cannot be determined from where a publisher on a particular topic/stream will publish from. Propagating subscription data throughout the broker network is the only way (besides sending all published messages to every broker) to guarantee that published messages, from wherever they may be published, will make their way to the subscribers who have requested the messages. This requirement imposes a great strain on the broker network, as it not only presents a high data traffic level throughout the network but also the subscription data must be locally stored and maintained with respect to each broker in the broker network.

SUMMARY OF THE INVENTION

According to one aspect, the present invention provides in a publish/subscribe data processing broker network having a plurality of broker data processing apparatuses each of which has an input for receiving published messages directly from a publisher application and/or receiving subscription data from a subscriber application, a first broker data processing apparatus comprising: means for receiving a data message published on a first topic by a first publisher application; and means for forwarding the received published data message to a subscriber application which has requested, by entering subscription data, to receive a message on the first topic; wherein the first broker data processing apparatus sends a declaration to at least one other broker data processing apparatus of said plurality of broker data processing apparatuses declaring that the first broker data processing apparatus is the only broker data processing apparatus that is directly communicating with a publisher application that is publishing on the first topic.

According to a second aspect, the present invention provides a data processing method having method steps corresponding to each element of the data processing apparatus of the first aspect of the invention.

According to a third aspect, the present invention provides a computer readable storage medium having a computer program stored on it which, when executed on a computer, carries out the functionality of data processing method of the second aspect of the invention.

The present invention allows one broker in a network of such brokers, to be declared as the unique source of taking published messages into the network for a particular topic.

Thus, with the present invention, since a publisher application can be declared as the unique source of publications on a stated topic in the network, the problem that existed in the prior art of requiring subscription data to be propagated, maintained and stored by each distribution agent throughout the broker hierarchy no longer exists. Specifically, the problem no longer exists because there is no more uncertainty regarding where a publisher application might publish from. Thus, subscription data need only be propagated to and maintained on distribution agents which are included in a direct path between the unique broker source on the stated topic and a subscriber which has subscribed to that topic.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention will be better understood by referring to the detailed description of the preferred embodiments which will now be described in conjunction with the following drawing figures:

FIG. **1** shows the architecture of a prior art publish/subscribe broker network which was referred to above; and

FIG. **2** shows the architecture of a publish/subscribe broker network according to which the preferred embodiment of the present invention will be explained below.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the prior art FIG. **1** discussed above, a publisher application **11**, running on one computer, is, for example, a supplier of live stock market data quotes. That is, publisher application **11** provides frequent messages stating the present value of share prices. In this example, publisher application **11** is publishing messages on a stream called "stock" which has already been configured in the broker network **2**. As is well known, when publisher **11** wishes to publish a stock quote message to stream "stock", publisher **11** makes an RPC call to the root distribution agent **11** which is at the top level of the broker network tree structure. In this example, subscriber application **32**, running on another computer, has sent a subscription request via an RPC call to leaf distribution agent **24**, which is at the bottom level of the tree structure, indicating that subscriber **32** would like to subscribe to stream "stock".

Thus, whenever publisher **11** publishes a data message to stream "stock" the distribution tree structure of broker network **2** channels the message down through the root distribution agent **21**, through any intermediary distribution agents (e.g., **22** in the example of FIG. **1**) and through the leaf distribution agent **24** to the subscriber **32**. This involves a series of RPC calls being made between each successive circle in the diagram of FIG. **1** connecting publisher **11** and subscriber **32** (i.e., **11** to **21**, **21** to **22**, **22** to **24** and **24** to **32**).

FIG. **2** shows a different publish/subscribe architecture where publisher applications can publish messages to the broker network by directly communicating with any one of a plurality of distribution agents (brokers). For example, publisher application **201** is shown communicating directly with Broker **12**. There is no requirement in this architecture that all publisher applications communicate directly with a top (or root) distribution agent. Publisher application **201** can potentially communicate directly with any of the distribution agents shown in FIG. **2**, in the described examples below it will be shown communicating directly with Broker **12**.

Subscriber applications **202** and **203** would like to receive messages on the stream/topic that publisher application **201** is publishing on. Thus, subscriber applications **202** and **203** communicate directly with Brokers **1112** and **1221**, respectively, to provide subscription data thereto informing the broker hierarchy of their desire to receive such published messages. Since the publisher application **201** is allowed to communicate directly with any of a plurality of distribution agents, the subscription data entered by the subscriber applications must be propagated throughout the broker network to each Broker shown in FIG. **2**. This way, no matter which distribution agent the publisher application **201** happens to communicate directly with, the published messages will be able to be routed to the subscriber applications **202** and **203**. As stated above, however, this creates a high performance overhead due to the excessive amounts of subscription data propagation traffic throughout the broker network and due to the need to have to maintain and store such subscription data locally at each distribution agent.

If a distribution agent (also referred to herein as a "broker") can be identified to the other distribution agents as the home to all publisher applications (e.g. by topic content or a publisher flag) on a given topic, call this a unique publisher broker for simplicity, it is possible to restrict the subscription path in the hierarchy by halting the propagation of the subscription data once this unique publisher broker is reached.

To further the limiting of subscription propagation in the unique publisher broker case it is possible to remove subscriptions that have been propagated down branches of the hierarchy leading off the path between the subscriber and the publisher that contain no subscriptions or the publisher on this topic, thus, reducing the subscriptions for a topic to only lie on the path(s) between the subscriber's (or subscribers') broker(s) and the publisher's broker.

The first level of subscription data propagation restriction prevents subscription data from flowing further once the

unique publisher broker is reached by the subscription data. When a subscription for a topic arrives at a unique publisher broker and the topic matches the topic on which this broker is the unique publisher broker, the unique publisher broker will not propagate the subscription any further through the hierarchy as it is known that no other broker can possibly publish on this topic. For example, if a new subscriber application **203** attaches to its nearest Broker **1221** and enters a subscription to a certain topic (e.g., IBM stock price), this subscription data identifying the new subscription will propagate up to Broker **122** and then further up to Broker **12** (which has previously declared itself to the other brokers as the unique publisher broker on the topic of IBM stock price). Broker **12** will then recognize that the subscription data's topic (IBM stock price) matches the topic (IBM stock price) on which Broker 12 is the unique publisher broker, and thus Broker **12** will not further propagate the subscription data to Broker **121** or Broker **1**.

The second level of limiting subscription data propagation is the removal of unnecessary subscriptions which has already been propagated to brokers, i.e., those subscriptions that do not lie on the path(s) between subscriber(s) and the unique publisher broker, once a new unique publisher broker is added to an existing broker hierarchy. Any unnecessary subscriptions can be identified by the fact that they would cause publications to flow in the opposite direction from those originating from the unique publisher broker, which is not possible for they would have to have originated from a publisher on another broker, and thus, the publisher broker could not be unique.

The preferred embodiment involves the use of a special message (for example, a publication), call it a unique publisher broker message, this contains the topic concerned and the identity of the broker that has just sent this message. A broker receiving a unique publisher message will follow these rules:

1) If this broker also claims to be a unique publisher broker on this same topic we have a situation where more than one broker in the hierarchy believe they are unique publishers on the same topic, this cannot be valid and an error is reported. Otherwise: The broker marks the topic that matches the one in the message as being a unique publisher topic.
2) If the broker has a subscription from the broker that sent this message, the subscription can be removed. This is because the subscription could only be used if a publication arrived at this broker and was to be propagated towards the broker sending the unique publisher message. This would cause publications to flow towards the publisher which is not possible when the publisher is unique. The identity of the broker sending this message is replaced with the identity of the current broker and the message is then propagated to every relation known to this broker, except the one that originated the unique publisher message.

Now we define the three scenarios that can cause a unique publisher message to be generated by a unique publisher broker and how they are handled:

1) Subscriber applications subscribe to a topic by communicating directly (e.g., via RPC) with one of the brokers, and the subscriptions (i.e., subscription data) are propagated to all brokers before a unique publisher has been identified. When a broker (e.g., Broker **12**) declares that it is the unique publisher broker on this topic and subscription(s) already exist, the unique publisher broker (e.g., Broker **12**) marks the topic as being unique and a unique publisher message is generated and sent to all relations (meaning, all brokers that are direct neighbours) of this broker (e.g., Brokers **121**, **122** and **1**). By following the above rules this message will be propagated to all brokers and any redundant subscriptions will be removed from the hierarchy.
2) Before any subscriptions are made, a publisher broker (e.g., broker **12**) believes that it is a unique publisher broker on a certain topic (e.g, IBM stock price). A subscription to this topic then arrives at broker **12** from another broker (e.g., broker **1**), once a subscriber application **202** has entered a subscription (e.g., by directly communicating the subscription data to broker **1112**, which has resulted in corresponding subscription data propagating to brokers **111**, **1111**, **11**, **112**, **1121**, **1** and finally to broker **12**). At this point (when the subscription data reaches broker **12**) we halt propagation of the subscription past broker **12**, and broker **12** generates a unique publisher message and sends it to the broker **1** that sent the subscription data to broker **12**. Again, by following the above rules this unique publisher message will be propagated from broker **1** to all brokers (i.e., **11**, **112**, **1121**, **111**, **1112** and **1111**) that have received the original subscription data. Then, the subscription data is removed from those brokers (i.e., **112**, **1121**, **1111**) lying off the direct path between the unique publisher broker **12** and the subscriber application **202**.
3) A unique publisher broker **12** exists along with subscriber **202** and a direct path (i.e., from subscriber **22** to broker **1112** to broker **111** to broker **11** to broker **1** to broker **12**) between them has been formed. Then, a new subscription (from a new subscriber **204**, shown in dotted line, is made from a broker **1121** that lies in a branch off a direct path from the unique publisher broker **12** to an existing subscriber **202**. When the new subscription data arrives at broker **11** (which is on the direct path mentioned above) and the topic of the subscription has been marked as a unique publisher topic and a subscription to this topic already exists it is now known that we have intercepted a direct path between a publisher and a subscriber. The propagation of the subscription is halted at broker **11** (i.e., the subscription data has already propagated from broker **1121** to broker **112** to broker **11**), as a subscription to this topic would already have been propagated from broker **11** to the unique publisher broker **12** due to the existing subscription. A unique publisher message is then generated by broker **11** and sent back to the broker **112** that sent the new subscription. This is the same as the scenario above, only for a sub-tree of the broker hierarchy.

While the preferred embodiment of the invention has been discussed in the context of a broker network made up of a hierarchy (e.g., designed from the top down) of distribution agents, the broker network need not be hierarchical. For example, the network could also be configured as a totally connected network, with each broker connected to every other broker (or some other combination of brokers less than every other broker).

We claim:

1. In a publish/subscribe data processing broker network having a plurality of broker data processing apparatuses each of which has an input for receiving published messages directly from a publisher application and/or receiving subscription data from a subscriber application, a first broker data processing apparatus comprising:

means for receiving a data message published on a first topic by a first publisher application; and

means for forwarding the received published data message to a subscriber application which has requested,

by entering subscription data, to receive a message on the first topic;

wherein the first broker data processing apparatus sends a declaration to at least one other broker data processing apparatus of said plurality of broker data processing apparatuses declaring that the first broker data processing apparatus is the only broker data processing apparatus that is directly communicating with a publisher application that is publishing on the first topic.

2. The apparatus of claim 1 wherein a second broker data processing apparatus, which is on a direct path between the first broker data processing apparatus and a subscriber application, sends the declaration on behalf of the first broker data processing apparatus upon receiving new subscription data from a new subscriber application to the first topic.

3. The apparatus of claim 1 wherein upon receipt of the declaration subscription data is removed from broker data processing apparatuses that do not lie on a direct path between the first broker data processing apparatus and the subscriber application.

4. The apparatus of claim 1 wherein the network is the Internet.

5. The apparatus of claim 1 wherein at least one of the publisher application and the subscriber application runs in cooperation with a World Wide Web browser application.

6. In a publish/subscribe data processing broker network having a plurality of broker processing apparatuses each of which has an input for receiving published messages directly from a publisher application and/or receiving subscription data from a subscriber application, a method carried out by a first broker data processing apparatus, the method comprising steps of:

receiving a data message published on a first topic by a first publisher application; and

forwarding the received published data message to a subscriber application which has requested, by entering subscription data, to receive a message on the first topic;

wherein the first broker data processing apparatus sends a declaration to at least one other broker data processing apparatus of said plurality of broker data processing apparatuses declaring that the first broker data processing apparatus is the only broker data processing apparatus that is directly communicating with a publisher application that is publishing on the first topic.

7. In a publish/subscribe data processing broker network having a plurality of broker data processing apparatuses each of which has an input for receiving published messages directly from a publisher application and/or receiving subscription data from a subscriber application, a computer program product embodied on a computer readable storage medium for, when run on a computer, carrying out a method on a first broker data processing apparatus, the method comprising steps of:

receiving a data message published on a first topic by a first publisher application; and

forwarding the received published data message to a subscriber application which has requested, by entering subscription data, to receive a message on the first topic;

wherein the first broker data processing apparatus sends a declaration to at least one other broker data processing apparatus of said plurality of distribution agent data processing apparatuses declaring that the first broker data processing apparatus is the only broker data processing apparatus that is directly communicating with a publisher application that is publishing on the first topic.

* * * * *

US005884031A

United States Patent [19]

Ice

[11] Patent Number: 5,884,031

[45] Date of Patent: Mar. 16, 1999

[54] METHOD FOR CONNECTING CLIENT SYSTEMS INTO A BROADCAST NETWORK

[75] Inventor: **Jeffrey L. Ice**, Loxahatchee, Fla.

[73] Assignee: **Pipe Dream, Inc.**, West Palm Beach, Fla.

[21] Appl. No.: **724,333**

[22] Filed: **Oct. 1, 1996**

[51] **Int. Cl.**[6] **G06F 13/00**

[52] **U.S. Cl.** **395/200.33**

[58] **Field of Search** 395/200.3, 200.33, 395/200.47, 200.57; 340/825.01

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,353,412 10/1994 Douglas et al. .

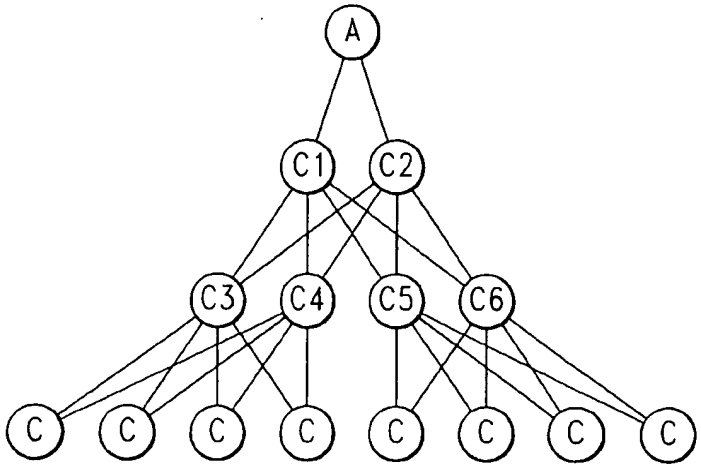
5,461,624 10/1995 Mozzola .

Primary Examiner—Moustafa M. Meky
Assistant Examiner—Hassan Ibrahim
Attorney, Agent, or Firm—Norman Friedland

[57] **ABSTRACT**

A private network is built by first allowing a pre-determined number of client systems to connect directly to a server system. After this occurs, additional client systems requesting connection are furnished with the addresses of client systems already connected within the private network. Each of the additional client systems then makes connections with a multiple number of client systems to receive information from the server system. Each of these client systems subsequently accepts connections from up to a second predetermined number of client systems to which it transmits information received from the server system.

15 Claims, 5 Drawing Sheets



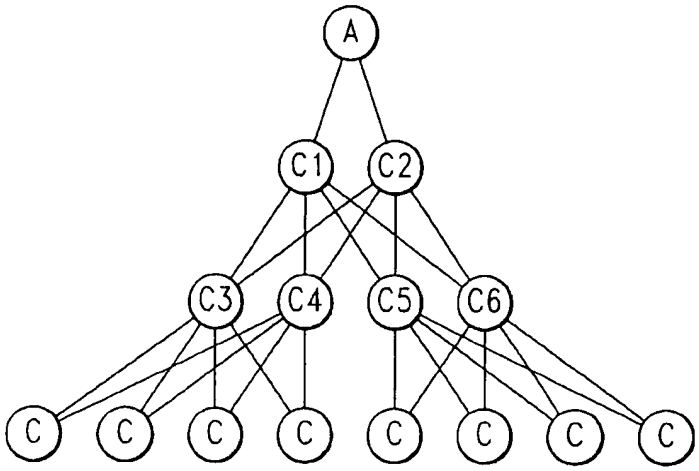

A
C1
C2
C3
C4
C5
C6
C
C
C
C
C
C
C
C

FIG. 1.

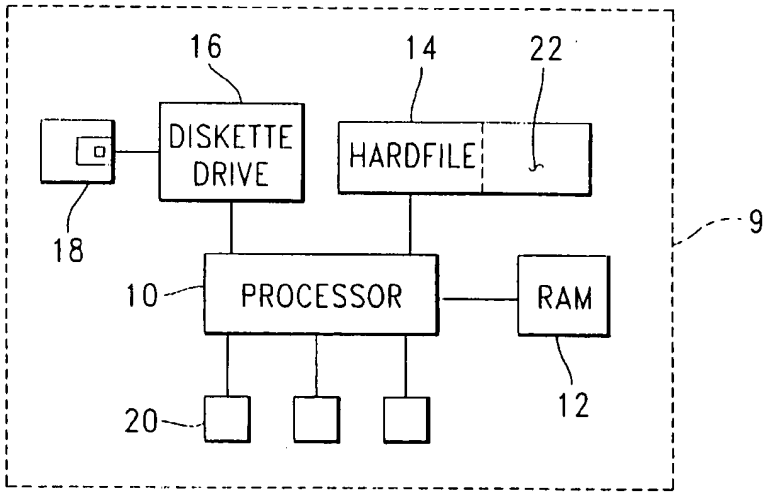

16
14
22
DISKETTE DRIVE
HARDFILE
18
9
10
PROCESSOR
RAM
20
12

FIG. 2.

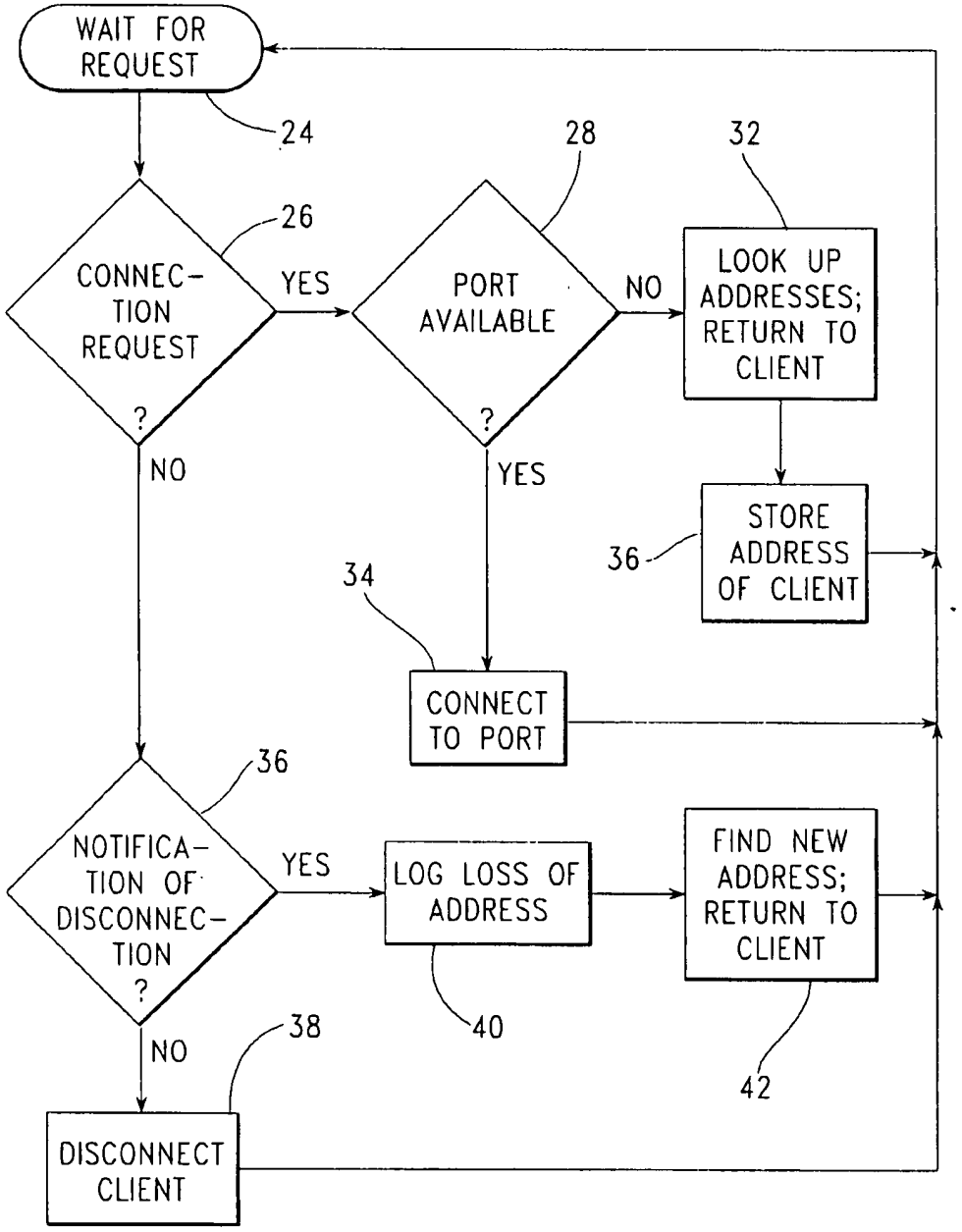

WAIT FOR REQUEST
24
CONNEC-TION REQUEST ?
26
YES
PORT AVAILABLE ?
28
NO
LOOK UP ADDRESSES; RETURN TO CLIENT
32
NO
YES
STORE ADDRESS OF CLIENT
36
34
CONNECT TO PORT
NOTIFICA-TION OF DISCONNEC-TION ?
36
YES
LOG LOSS OF ADDRESS
40
FIND NEW ADDRESS; RETURN TO CLIENT
42
NO
38
DISCONNECT CLIENT

FIG. 3.

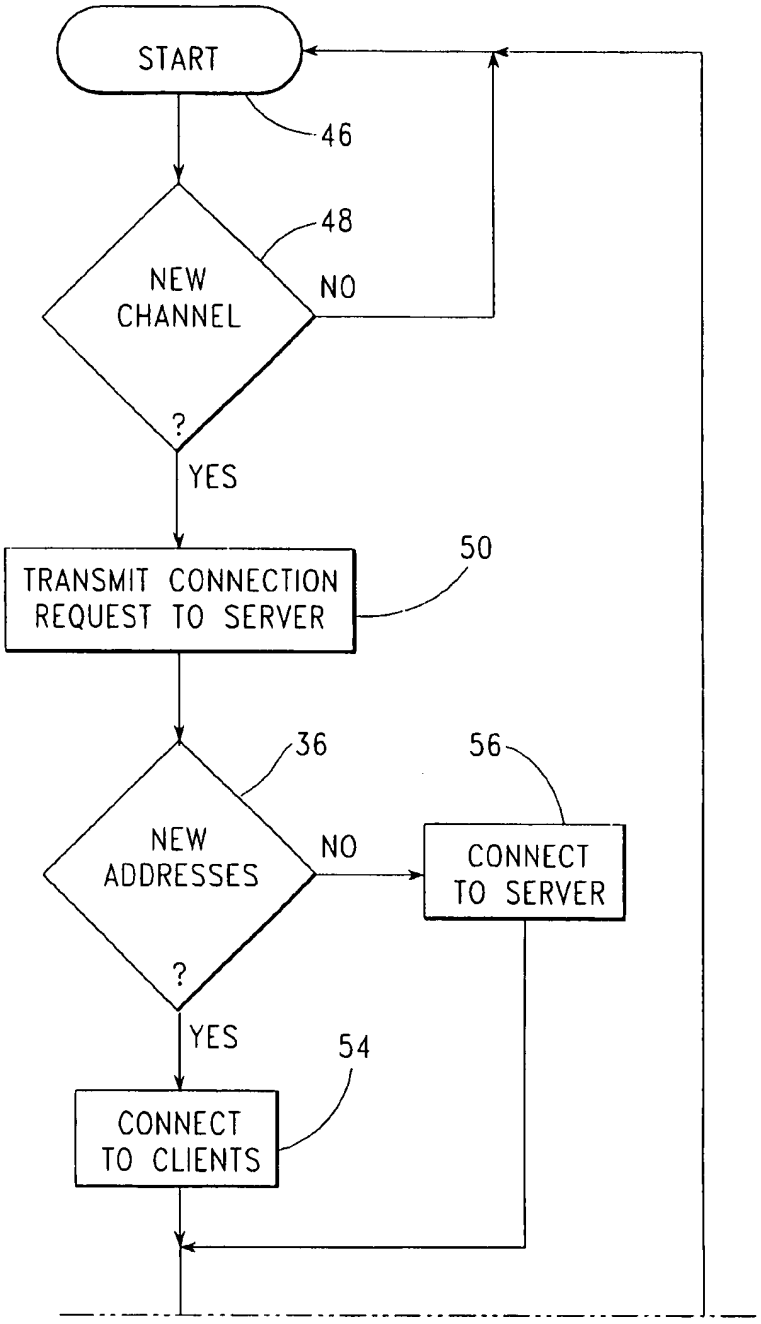


FIG. 4A.

FIG. 4A.
FIG. 4B.

FIG. 4.

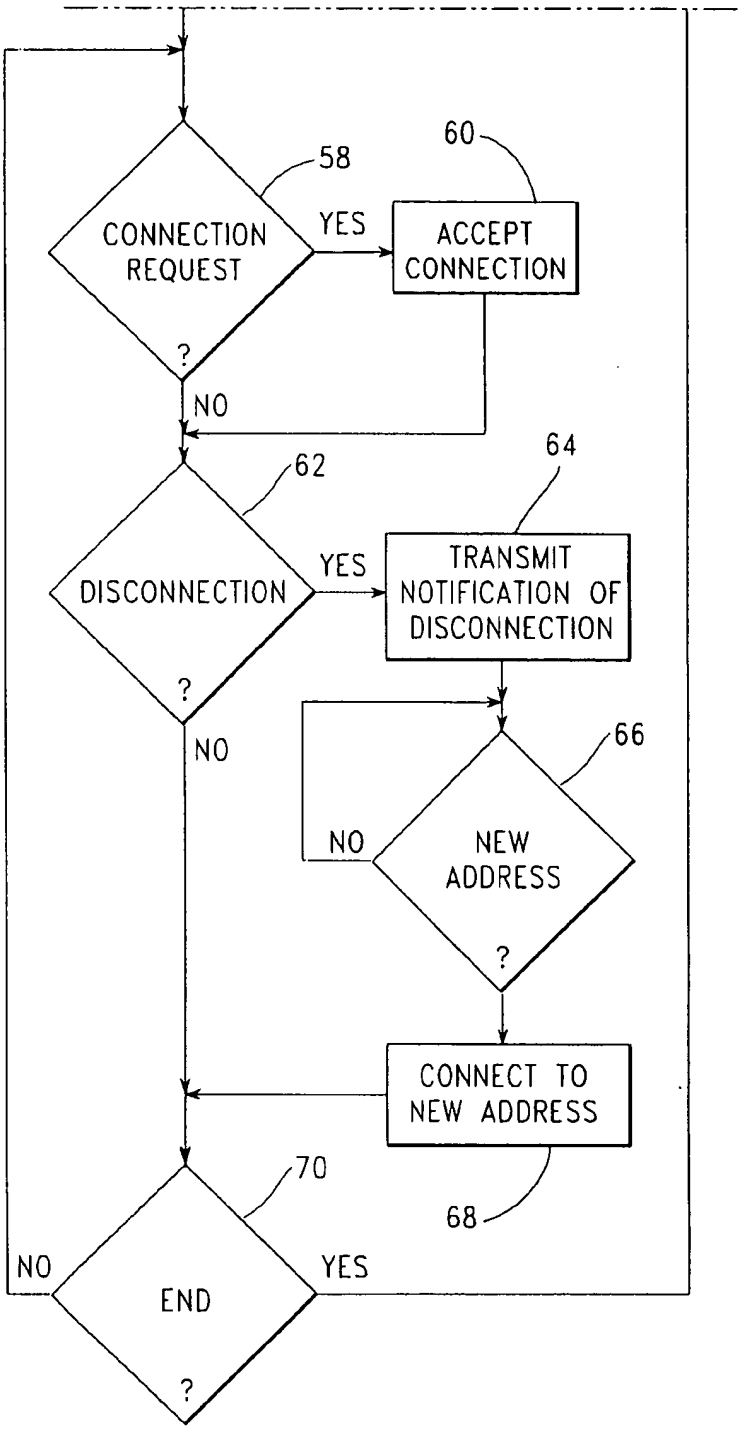

58
60
CONNECTION REQUEST
YES
ACCEPT CONNECTION
?
NO
62
64
DISCONNECTION
YES
TRANSMIT NOTIFICATION OF DISCONNECTION
?
NO
66
NO
NEW ADDRESS
?
CONNECT TO NEW ADDRESS
68
70
NO
END
YES
?

FIG. 4B.

NUMBER OF ROWS	CLIENT SYSTEMS IN LAST ROW	TOTAL CLIENT SYSTEMS
1	2	2
2	4	6
3	8	14
4	16	30
5	32	62
6	64	126
7	128	254
8	256	510
9	512	1,022
10	1,024	2,046
11	2,048	4,094
12	4,096	8,190
13	8,192	16,382
14	16,384	32,766
15	32,768	65,534
16	65,536	131,070

FIG. 5.

METHOD FOR CONNECTING CLIENT SYSTEMS INTO A BROADCAST NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a method for making connections on the Internet, and, more particularly, to a method for causing connections to be made among client systems.

2. Background Information

The conventional structure of the Internet is based on combinations of clients and servers, in which the client systems obtain information from the servers. Thus, to retrieve information, an individual client system makes a connection with a server and requests the particular information needed. The information is then sent from the server to the client. With this method, each client has an individual connection to the server. If there are too many connections to the server to allow a new connection to be established, a client wishing to make an additional connection is denied access. What is needed is a way to increase the number of clients which can be connected to a single server without requiring a substantial increase in the server hardware to provide additional ports.

Some servers provide what appears to be a broadcasting function by relaying information from one client connected to the server to all other clients connected to the same server. An example of this type of information sharing is found in the "chat relay" server protocol. However, this type of interconnection is also limited by the number of clients which can be connected to the server. Again, what is needed is a way to increase the number of clients which can be interconnected in a substantial way.

SUMMARY OF THE INVENTION

In accordance with one aspect of the invention, there is provided a process for connecting client systems within a private network. This process includes the steps of connecting client systems, up to a first pre-determined maximum number of systems, directly to a server system through a public network, and connecting additional client systems, beyond the first pre-determined maximum number, to other client systems to form the private network extending through connections within the public network to the server system.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a diagrammatic view of a private network built into a pyramid in accordance with the present invention;

FIG. 2 is a block diagram of a computing system, representing either a server system or a client system in FIG. 1;

FIG. 3 is a flow chart of processes occuring in the server system of FIG. 1 to provide for the connection of client systems into the private network of FIG. 1 or for their disconnection therefrom;

FIGS. 4A–4B is a flow chart of processes occuring in the server system of FIG. 1 to provide for the connection of client systems into the private network of FIG. 1 or for their disconnection therefrom, with FIG. 4A being an upper portion of FIG. 4, and with FIG. 4B being a lower portion thereof; and

FIG. 5 is a table showing the number of systems in a last row of the network of FIG. 1, and the total number of systems in the network, as a function of the number of rows therein.

DETAILED DESCRIPTION

FIG. 1 is a diagrammatic view of a private network built into a pyramid in accordance with the present invention, showing connections established among a single server and various clients. In this diagram, the single server is indicated as A, and the clients are indicated as C, followed by numerals as required for identification.

The establishment of this network structure is begun as the client C1 requests a connection to the private network by sending a message to server A. Since a connection can be made directly the server A, with client C1 being the first client to connect, the server A responds by telling the client C1 to establish a direct connection to the server. The same procedure if followed when a second client C2 requests a connection to the private network.

However, in the example of FIG. 1, only two connections can be made directly to the server A. Therefore, when a new client C3 requests a connection from the server A, it receives an instruction from server A to connect to client C1, and also to C2 in case C1 goes down. Similarly, additional clients C4, C5, and C6 are connected to clients C1 and C2.

The expanding structure of the private network is determined by the number of clients which are allowed to connect to each client. In the example of FIG. 1, each client, except for C1 and C2, is connected upward to two clients, and each client, except for those at the bottom, or end, of the structure, is connected to four clients below. Thus, with these exceptions, each client is connected to two additional clients for receiving information, so that continuity is maintained if a client goes down, and up to four clients for disseminating information. The server A sends information from a database of data to be broadcast to the clients to which it is directly connected, in this example to clients C1 and C2. These clients, and those below them, in turn relay the data from one level to another.

FIG. 2 is a block diagram of a computing system, which represents both the server A of FIG. 1 and ono of the client systems C, also shown in FIG. 1. This computing system 9 includes a processor 10, associated system memory 12, a hardfile 14, a disk drive 16 for reading a floppy disk 18, and a number of output ports 20 for connections with client systems.

The program connecting server A with these client systems in accordance with this invention is preferably written on one or more diskettes 18, to be loaded within the server A into system memory 12 and hardfile storage 14 through the disk drive 16. The hardfile 14 also includes a database 22 holding a list of all clients presently connected to the private network, with their IP (Internet Protocol) addresses. While this database 22 is shown as residing in the hardfile 14, it may reside additionally, partially, or alternately in system memory 12, facilitating rapid access to the address information.

The program connecting a client system C to server A through other client systems C in accordance with this invention is loaded within the client system C into system memory 12 and hardfile 14 from one or more diskettes 18. Alternately, this program may be downloaded through network connections between the server A and the client C into the system memory 12 of the client C for execution by the processor 10 and for storage within the hardfile 14.

FIG. 3 is a flow chart of processes occurring within the server A (shown in FIG. 1) operating in accordance with the present invention to provide for the connection of client systems to the private network and for their disconnection therefrom.

Referring to FIGS. **2** and **3**, in block **24**, the server A is waiting for an incoming request from a client, which may be either a request for a new connection or a notification from a disconnection of an existing connection. A notification of disconnection is sent to server A by a system which has been connected to the system disconnecting from the private network, but which is one connection farther away from the server than the disconnecting system. Thus, the system notifying the server of a disconnection has just lost one of its two private network connections for receiving information from server A.

When a request is received, a determination is made in block **26** of whether it is a request for connection. If the request is a request for connection, a determination is made in block **28** of whether a port **20** of the server A is available. In the example of FIG. **1**, the connections of clients C1 and C2 were made directly to the server because a port **20** was available when the connection request was made. Thus, if a port **20** is available, as determined in block **28**, the client making the request is connected to the port in block **30**. If a port **20** is not available, in block **32**, the server A looks up two available IP addresses within its database **22** and returns this information to the requesting client. Next, in block **34**, the server A logs the IP address of the client which has made the request into the database **22**, so that additional clients may subsequently be connected to this new element within the private network. Then the server A returns to block **24** to wait for the next request.

On the other hand, if the request to server A is determined in block **26** not to be a request for connection, a determination is made in block **36** of whether it is a notification of a disconnection. If it is not such a notification, in block **38**, the server A disconnects the client system making the request from the private network, as it has made an invalid request. If it is determined in block **36** that the request is a notification of disconnection, in block **40** the address of the disconnected client is removed from database **22**. Next, in block **42**, server A looks up an additional available IP address within the database **22** and returns this information to the client providing notification of a disconnection, so that it can re-establish a second data link through a new client whose address has been provided. Then the server returns again to block **24** to wait for another request to process.

FIG. **4** is a flow chart of processes occurring within the client C (shown in FIG. **1**) operating in accordance with the present invention to provide for the connection of client systems to the private network and for their disconnection therefrom.

Referring to FIGS. **1** and **4**, these processes begin with the system in start block **46**, waiting for a request for a new connection. In start block **46**, the system may be, for example, performing calculations or providing other functions for its user. The command to make a new connection may come from a user interaction with the system. After this command occurs, as determined in block **48**, a request for a connection is sent to the server A in block **50**. Then the server connects the client system directly to itself, if a port for such a connection is available, or sends two new addresses. If the new addresses are sent, as determined in block **52**, connection is made to the new addresses in block **54**. Otherwise, a direct connection is made to the server in block **56**.

In either case, the server A can now instruct up to four other client systems C to connect to the client executing the processes of FIG. **4**. Thus, when such a connection request is received from another client, as determined in block **58**, the connection is accepted in block **60**. If one of the two systems to which the client C is connected becomes disconnected for any reason, as determined in block **62**, the client C transmits a disconnection notification to the server A in block **64**. When the server subsequently responds with a new address, as determined in block **66**, a connection is made to the new address in block **68**. Connection within the private network may be terminated, for example, at the request of the user, at any time. When this occurs, as determined in block **70**, the client system returns to the start in block **46**.

FIG. **5** is a tabular view of the number of client systems C which may be connected in the last row (i.e. the row farthest from the server A), together with the total number of client systems C as a function of the number of client system rows in the private network structure. In general, the last row is only partly filled, with a total number of client systems other than one of the total numbers listed in FIG. **5**.

The table of FIG. **5** reflects the assumptions of FIG. **1**, that two client systems are directly connected to the server system, that each other client system is connected to receive information from the server system through two other client systems, and that each client system, except for the last row thereof, has four other client systems connected to it to receive such information. While these pre-determined rules for connection are used to form the structure of the private network, they may be changed to vary the structure of the private network, or to account for various hardware configurations, within the scope of the present invention.

While the invention has been described in its preferred form or embodiment with some degree of particularity, it is understood that this description has been given only by way of example and that numerous changes in the details of construction, fabrication and use, including the combination and arrangement of parts and process steps, may be made without departing from the spirit and scope of the invention.

What is claimed is:

1. A process for connecting client systems within a private network, wherein said process comprises the steps of:

(a) connecting client systems, up to a first pre-determined maximum number thereof, directly to a server system through a public network;

(b) connecting additional client systems within a plurality thereof, beyond said first pre-determined maximum number thereof, to other client systems to form said private network extending through connections within said public network to said server system;

wherein each client system within said plurality thereof is connected to a second pre-determined number of client systems within said plurality thereof through said public network for receiving information from said server system;

wherein each said client system is connectable to up to a pre-determined number of client systems for transmitting information through said public network from said server system;

Wherein said step (b) includes the steps of:

(d) transmitting a request for connection from an additional client system within said plurality thereof to said server systems;

(e) transmitting addresses, of said second pre-determined number of available client systems connected within said private network, to said additional client system from said server system; and

(f) establishing connections between said additional client system and said second pre-determined number of available client systems connected within said private network.

2. The process of claim 1, additionally comprising a step (c) of maintaining said second pre-determined number of connections between each client system within said plurality thereof and other client systems within said plurality thereof, for receiving information from said server system by establishing new connections among said client systems when a disconnection occurs.

3. The process of claim 1, wherein each available client system within said second pre-determined number thereof has connected thereto for receiving information therefrom fewer than a third pre-determined number of client systems within said plurality thereof.

4. The process of claim 2, wherein said step (c) includes the steps of:

(g) transmitting a disconnection notification to said server system from a partly disconnected client system within said plurality thereof, wherein said partly disconnected client system has been connected for receiving information from said server system to a client system disconnecting from said private network;

(h) transmitting, from said server system to said partly disconnected client system, an address of an available client system within said plurality thereof connected within said private network; and

(i) establishing a connection between said available client system and said partly disconnected client system.

5. The process of claim 4, wherein said available client system has connected thereto for receiving information therefrom fewer than a third pre-determined number of client systems within said plurality thereof.

6. A process executing within a server system for establishing a private network of client systems within a plurality thereof to receive information from said server system, wherein said process comprises the steps of:

(a) receiving a request for connection from a client system issuing said request;

(b) determining if a port of said server system is available for connection with said private network;

(c) if a port of said server system is available for connection within said private network, as determined in step (b), connecting said client system issuing said request to said port;

(d) if a port of said server system is not available for connection within said private network, as determined in step (b), sending to said client system issuing said request a first pre-determined number of addresses of available client systems within said plurality thereof connected within said private network; and

(e) adding an address of said client system issuing said request to a list of addresses within a database residing in said server system.

7. The process of claim 6, comprising in addition the steps of:

(f) receiving a disconnection notification from a partly disconnected system within said private network, wherein said partly disconnected client system has been connected for receiving information from said server system to a client system disconnecting from said private network;

(g) sending to said client system issuing said disconnection notification an address of a client system within said plurality thereof connected within said private network; and

(h) removing an address of said client system disconnecting from said private network from a database including addresses of client systems connected within said private network.

8. The process of claim 7, wherein, within said steps (d) and (g) said available client systems have fewer than a second predetermined number of client systems attached thereto for transmitting information received from said server system.

9. A process executing within a particular client system for establishing a private network of client systems within a plurality thereof to receive information from a server system, wherein said process comprises the steps of:

(a) transmitting a connection request to said server system;

(b) receiving addresses of available client systems within said private network from said server system;

(c) connecting to said available client system for receiving information from said server system;

(d) receiving a connection request from an additional client system within said plurality thereof; and

(e) establishing a connection with said additional client system to transmit information received from said server system.

10. The process of claim 9, additionally comprising the steps of:

(f) being disconnected from a first client system connected within said private network between said particular client system and said server system;

(g) transmitting a notice of disconnection to said server system;

(h) receiving an address of an available client system connected within said private network from said server system; and

(i) establishing a connection with said available client system identified by said address received in step (h).

11. A computer readable medium upon which coded steps are written for a process executing within a server system for establishing a private network of client systems within a plurality thereof to receive information from said server system, wherein said process comprises the steps of:

(a) receiving a request for connection from a client system issuing said request;

(b) determining if a port of said server system is available for connection with said private network;

(c) if a port of said server system is available for connection within said private network, as determined in step (b), connecting said client system issuing said request to said port;

(d) if a port of said server system is not available for connection within said private network, as determined in step (b), sending to said client system issuing said request a first pre-determined number of addresses of available client systems within said plurality thereof connected within said private network; and

(e) adding an address of said client system issuing said request to a list of addresses within a database residing in said server system.

12. The medium of claim 11, wherein said process comprises in addition the steps of:

(f) receiving a disconnection notification from a partly disconnected system within said private network, wherein said partly disconnected client system has been connected for receiving information from said server system to a client system disconnecting from said private network;

(g) sending to said client system issuing said disconnection notification an address of a client system within said plurality thereof connected within said private network; and

(h) removing an address of said client system disconnecting from said private network from a database including addresses of client systems connected within said private network.

13. The medium of claim **12**, wherein, within said steps (d) and (g) said available client systems have fewer than a second predetermined number of client systems attached thereto for transmitting information received from said server system.

14. A computer readable medium upon which coded steps are written for a process executing within a particular client system for establishing a private network of client systems within a plurality thereof to receive information from a server system, wherein said process comprises the steps of:

(a) transmitting a connection request to said server system;

(b) receiving addresses of available client systems within said private network from said server system;

(c) connecting to said available client system for receiving information from said server system;

(d) receiving a connection request from an additional client system within said plurality thereof; and

(e) establishing a connection with said additional client system to transmit information received from said server system.

15. The medium of claim **14**, wherein said process additionally comprises the steps of:

(f) being disconnected from a first client system connected within said private network between said particular client system and said server system;

(g) transmitting a notice of disconnection to said server system;

(h) receiving an address of an available client system connected within said private network from said server system; and

(i) establishing a connection with said available client system identified by said address received in step (h).

* * * * *